# PWIPPER - Projet Ruby on Rails

## LP CASIR 2020 -2021


### JEANTET Joey - REVELARD Raphaël / Binôme 2


## Environnement nécessaire

- Ruby: 3.0.0
- Rails: 6.1.3
- Node: 14.14.0 ou +

## Installation

Cloner le projet en local:
`git clone https://github.com/jojojojojota/pwipper.git`

Installer le projet:

`yarn install`
`bundle install`
`rake db:migrate`
`rake db:seed`

Lancer un serveur local:
`bin/rails server`

Se rendre sur: http://127.0.0.1:3000

## Tests

Lancer la commande: `rails test model -v`

## Fonctionnement global de l'application

Le but de ce projet est la prise en main du framework Ruby on Rails au travers de la création d'un simili-twitter.

Le site se nomme donc Pwipper, on y publie des pweeps (=tweets) avec un message et des pourtags (équivalent hashtags) réalisés avec le signe %.

#### Bandeau

Composé de trois boutons:

- *Home* pour aller sur la page principale (possible uniquement si connecté).
- *Register* pour enregistrer de nouveaux utilisateurs.
- *Login* pour choisir un profil à utiliser et accéder à son espace personnel.

#### Page de connexion - /login

On a le choix parmi les utilisateurs existants, sélectionnables en cliquant dessus.

**Création d'utilisateur - /utilisateurs/new**

On y accède en cliquant sur *Register* dans le bandeau.

**Page principale - /index ou /**

Affichage de l'espace personnel de l'utilisateur sélectionné, avec:

- à gauche: la liste des utilisateurs. Ils peuvent être ajoutés ou retirés de la liste des personnes suivies en cliquant sur le bouton en forme adjacent.
- au centre, de haut en bas: les informations de l'utilisateur, l'interface de création des pweeps et enfin l'affichage de ses propres pweeps ainsi que ceux des utilisateurs suivis.
- à droite: la liste des pourtags les plus populaires (10 maximums)

## Organisation du binôme

Nous avons fait un développement orienté fonctionnalités avec la répartitions suivantes:

- Joey: parties pweeps, authentification et pourtags (vue, modèle et contrôleur), tests
- Moi: partie follow (vue, modèle et contrôleur), vue des listes des utilisateurs et des pourtags, interface de login, relations des modèles, tests

Le reste à été fait en commun.

# Avis Raphaël REVELARD

Pour ma part j'ai eu du mal à maîtriser ce framework, ne comprenant pas toujours pourquoi ceci marchait ou non.
Il faut dire que je viens du monde objet et embarqué et ne suis jamais vraiment à l'aise avec la logique web.
Mais à titre de comparaison, j'ai préféré le framework Symfony (aussi pour sa documentation).

# Avis Joey JEANTET

Pour moi le ruby n'est en soit pas un mauvais langage, même plutôt agréable à utiliser malgré une syntaxe qui se veut lisible mais qui peut facilement perturber un développeur habitué par la syntaxe habituelle des langages grand public.

Par contre je trouve que ruby on rails est un très mauvais framework pour un développeur qui n'est pas spécialisé dedans, mais pourquoi je dis ça ?

**Pourquoi j'aime pas RoR**

**L'abstraction**

Premièrement effectivement ror simplifie énormément la vie dans plein de concept (objet, route, vue, ...) mais voilà problème c'est qu'en faisant trop simple on se retrouve avec une courbe d'apprentissage ignoble, le framework est beaucoup trop abstrait par rapport a un concurrent comme Laravel, Express, ou encore Wald (oui oui le miens), et on se retrouve dans la même situation que Magento de adobe, le framework est tellement conçu dans l'optique de faciliter la vie qu'il en devient compliqué a utiliser et nécessite des compétences spécifiques au framework (ce qui est dommage pour un outils avec le but de se simplifier la vie et gagner du temps).

**L'automatisme**

Un chose qui ma beaucoup perturbé sur ror c'est les méthodes transparentes, quand je crée un contrôleur de vue je m'attend à avoir la main sur la totalité du contrôleur pour rendre ma vue quand je le décide et non quand le framework l'a prévu. J'ai parfois été handicapé par cette fonctionnalité qui pour moi est une aberration qui nous fait économiser au grand max 14 caractères. Une petite méthode self.render m'aurait bien plus par exemple.

**Les routes**

On reste sur le même problème d'abstraction, quand j'ai du créer les routes pour la création des utilisateurs j'ai eu énormément de mal avec le système de formulaire et de route ressource. Je trouve qu'il y a beaucoup trop de type de formulaire possible et j'aurais préféré avoir à coder un peu plus mais avoir moins d'abstraction.

**Pourquoi finalement ma capillarité n'a pas blanchi**

**Les modèles**

J'ai trouvé l'approche des modèles très sympathique, pas besoin d'écrire de méthode par défaut, validation automatique, relations très simple a gérer, suppression en cascade quasiment transparente, en bref j'aime beaucoup l'approche des modèles.

**Conclusion**

En bref, je vais rester dans ma zone de confort et continuer d'utiliser des langages comme : c, c++, rust, go, js(ts), kotlin ou java qui sont plus complexe mais bien plus facile dans la personnalisation de sa méthode de travail.

**J'ai apprécier travailler sur le projet rails mais personnellement si je devais je n'utiliserais absolument pas ce framework pour un projet personnel ou professionnel.**